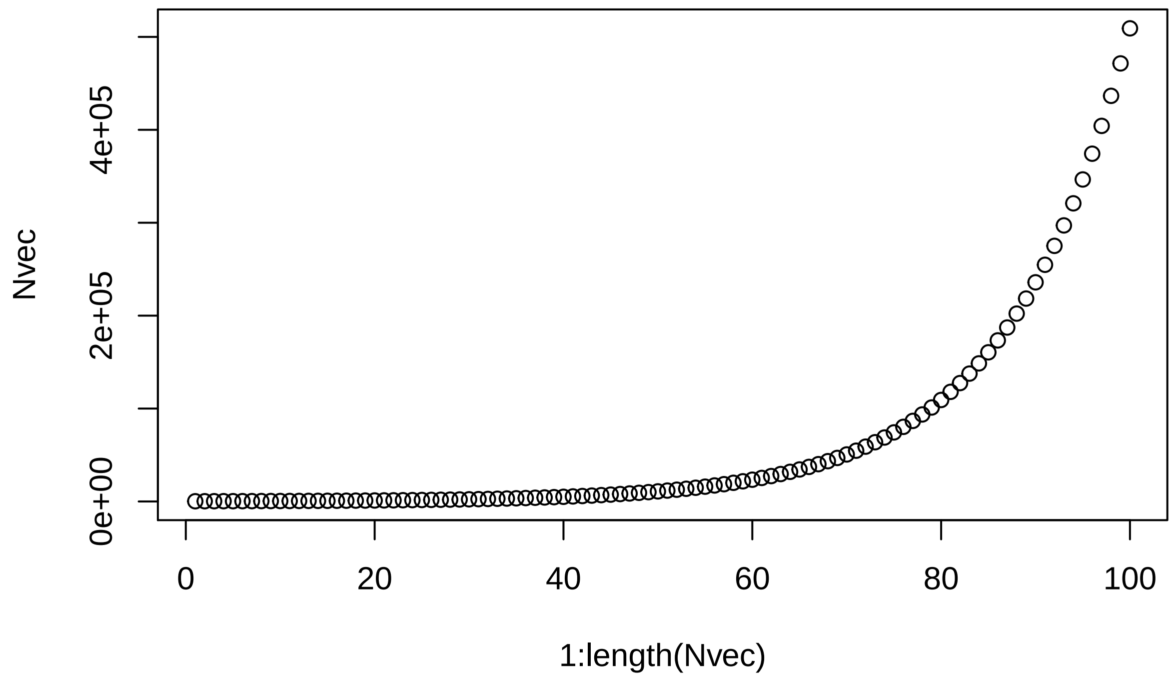
GrowthDynamics.R

Administrator

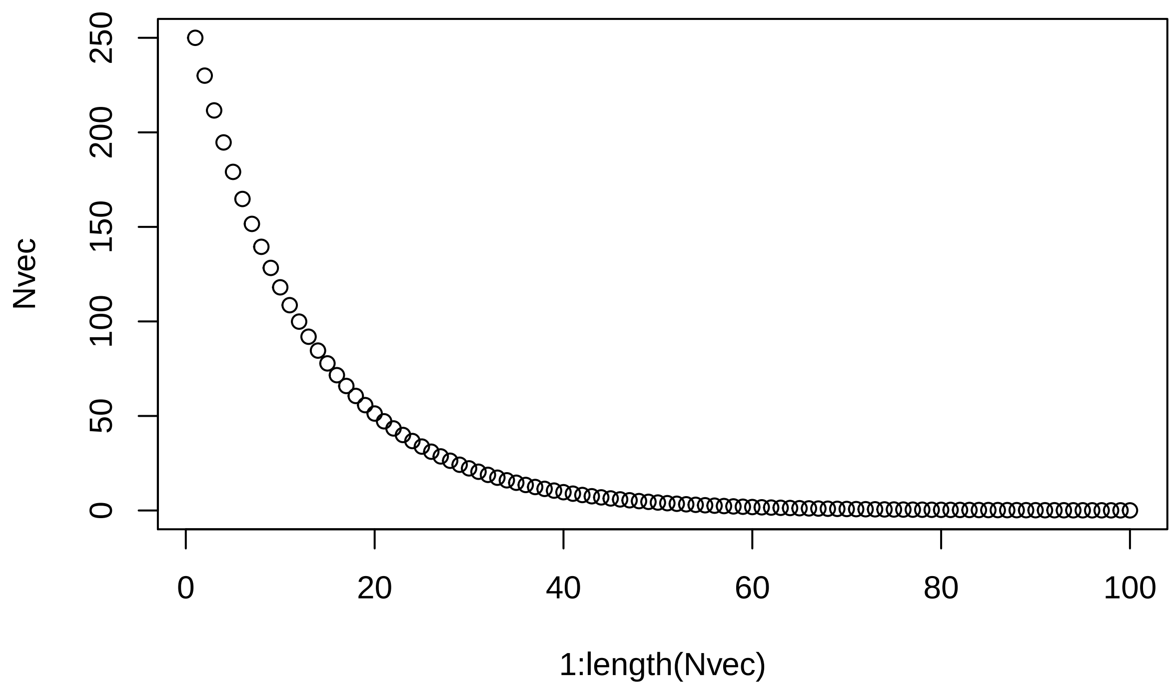
Thu Feb 25 15:11:14 2016

```
# Modeling differential growth equations
# February 22, 2016
# NJG

# basic equation for change plus current value
# illustrated with familiar exponential growth

rm(list=ls())
set.seed(100)
opar <- par(no.readonly=TRUE)

# basic growth model
```

```
Model1 <- function(N_0=250,Time=100,r=0.08){
  Nvec <- rep(0,Time) # create empty vector for length Time
  Nvec[1] <- N_0      # set initial value from input parameter
  for (i in 2:Time) {
    Nvec[i] <- Nvec[i-1] + r*Nvec[i-1] # basic change equation
  }
plot(x=1:length(Nvec),y=Nvec) # simple plot
  return(Nvec) # return the vector to the user

}
x <- Model1()
```




```
x <- Model1(r=-0.08)
```




```r
# sensitivity to time-step

r <- 0.08
BigStep <- seq(1,50,by=1)
SmallStep <-seq(1,50,by=0.1)
BigVec <- rep(0,length(BigStep))
SmallVec <- rep(0,length(SmallStep))
BigVec[1] <- 250
SmallVec[1] <- 250
for (i in 2:length(BigVec)) BigVec[i] <- BigVec[i-1] + r*BigVec[i-1]
for (i in 2:length(SmallVec)) SmallVec[i] <- SmallVec[i-1] + r*SmallVec[i-1]*0.1
tail(BigVec)
```

```
## [1]  7388.993  7980.112  8618.521  9308.003 10052.643 10856.855
```

```r
tail(SmallVec)
```

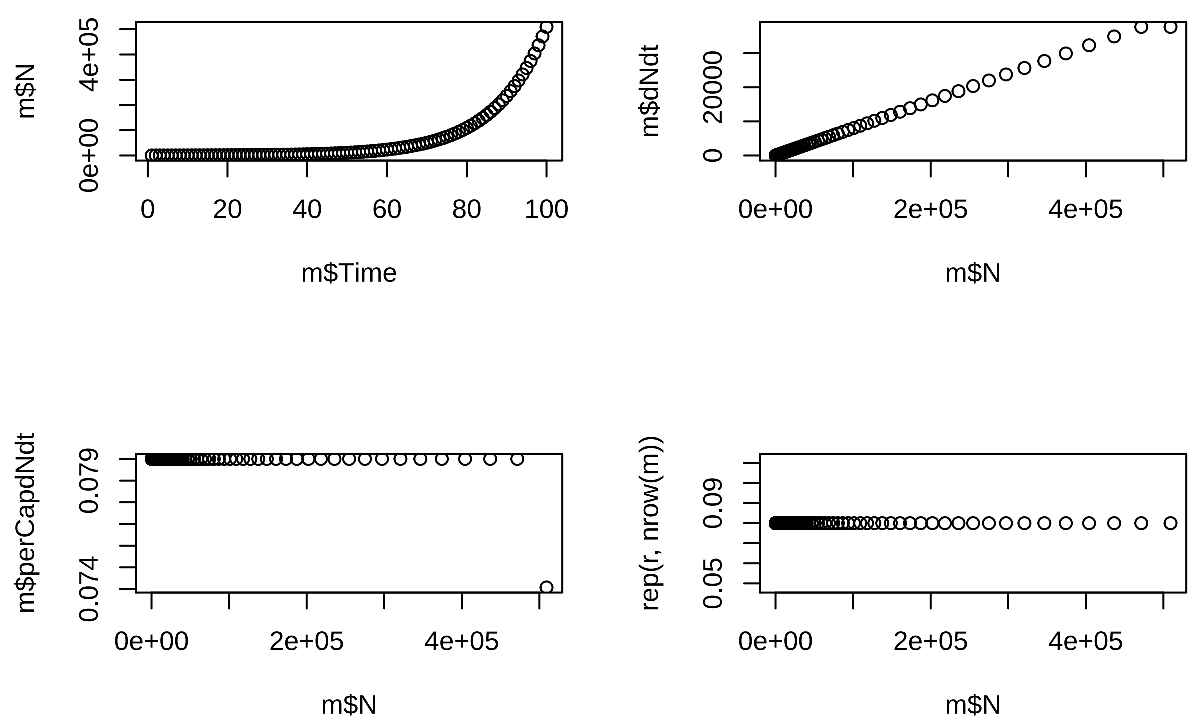
```
## [1] 11920.60 12015.96 12112.09 12208.99 12306.66 12405.11
```

```r
250*exp(50*0.08)
```

```
## [1] 13649.54
```

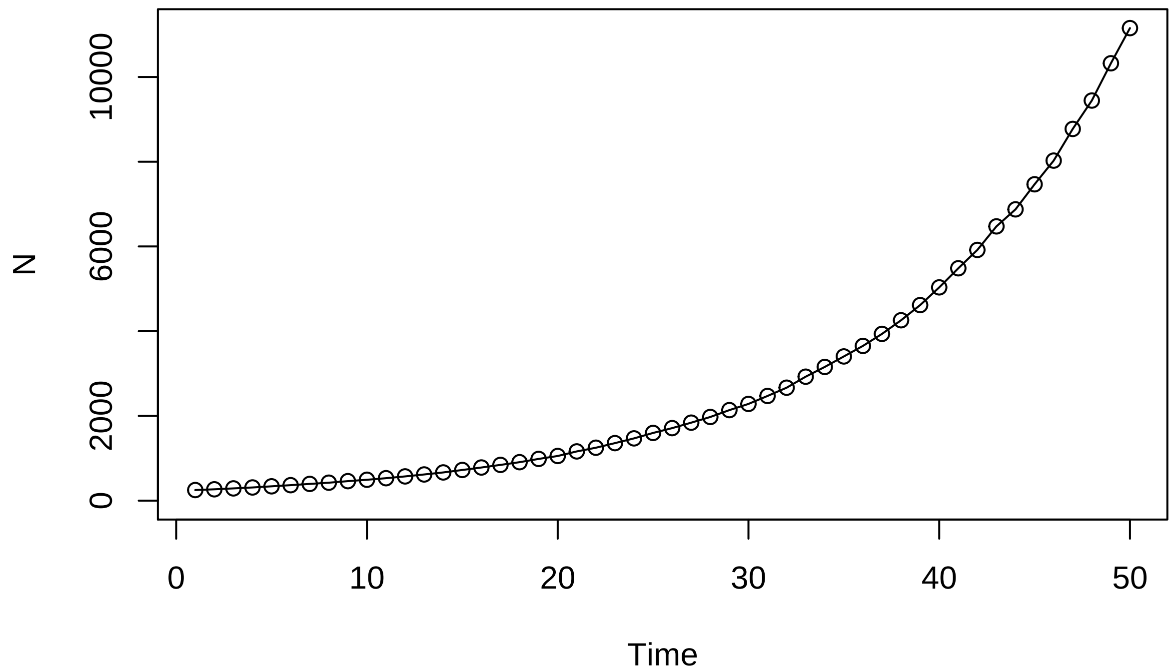
```r
# Calculating N, dNdt, (1/N)(dNdt)
```

```r
Model2 <- function(N_0=250,Time=100,r=0.08){
  Nvec <- rep(0,Time)
  Nvec[1] <- N_0
  for (i in 2:Time) {
    Nvec[i] <- Nvec[i-1] + r*Nvec[i-1] # exponential growth
  }
    dNvec <- diff(Nvec) # change in each time step
    dNvec <- c(dNvec,dNvec[length(dNvec)]) # repeat the last value
    perCapvec <- dNvec/Nvec # divide by N for per capita (1/N)(dN/dt)
m <- cbind(1:Time,Nvec,dNvec,perCapvec) # bind the 4 columns in a matrix
colnames(m) <- c("Time","N","dNdt","perCapdNdt")
return(m)
}

m <- Model2()          # save output matrix
m <- as.data.frame(m)  # convert to data frame
par(mfrow=c(2,2))      # set up a 2 x 2 plotting space
plot(x=m$Time,y=m$N)
plot(x=m$N,y=m$dNdt)
plot(x=m$N,y=m$perCapdNdt)
plot(x=m$N,y=rep(r,nrow(m)))
```



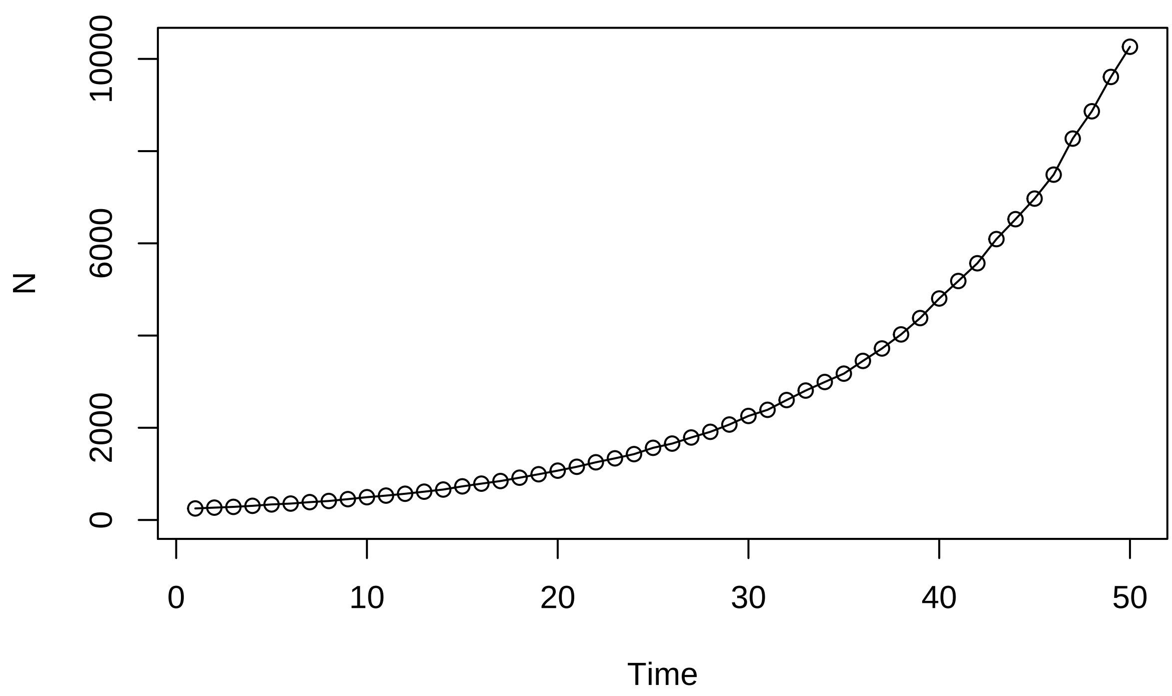
```r
par(opar)              # restore graphics parameters

# adding process error
```

```r
Model3 <- function(N_0=250,Time=50,r=0.08,sdR=0){

  Nvec <- rep(0,Time) # create empty vector of length Time
  Nvec[1] <- N_0 # set initial value from input parameter
  r <- rnorm(n=Time,mean=r,sd=sdR) # create a vector of random normal r values

    for (i in 2:Time) {
    Nvec[i] <- floor(Nvec[i-1] + r[i-1]*Nvec[i-1]) # floor for integers
    if(Nvec[i]<1) {
      Nvec[i] <- 0 # values < 1 (including negatives) set to 0
      break    # extinction at 0; exit the loop
    }
    }
plot(x=1:length(Nvec),y=Nvec,type="o",xlab="Time",ylab="N",ylim=c(0,max(Nvec)))
  return(Nvec)

}
Nvec <- Model3(r=0.08,sdR=0.01)
```



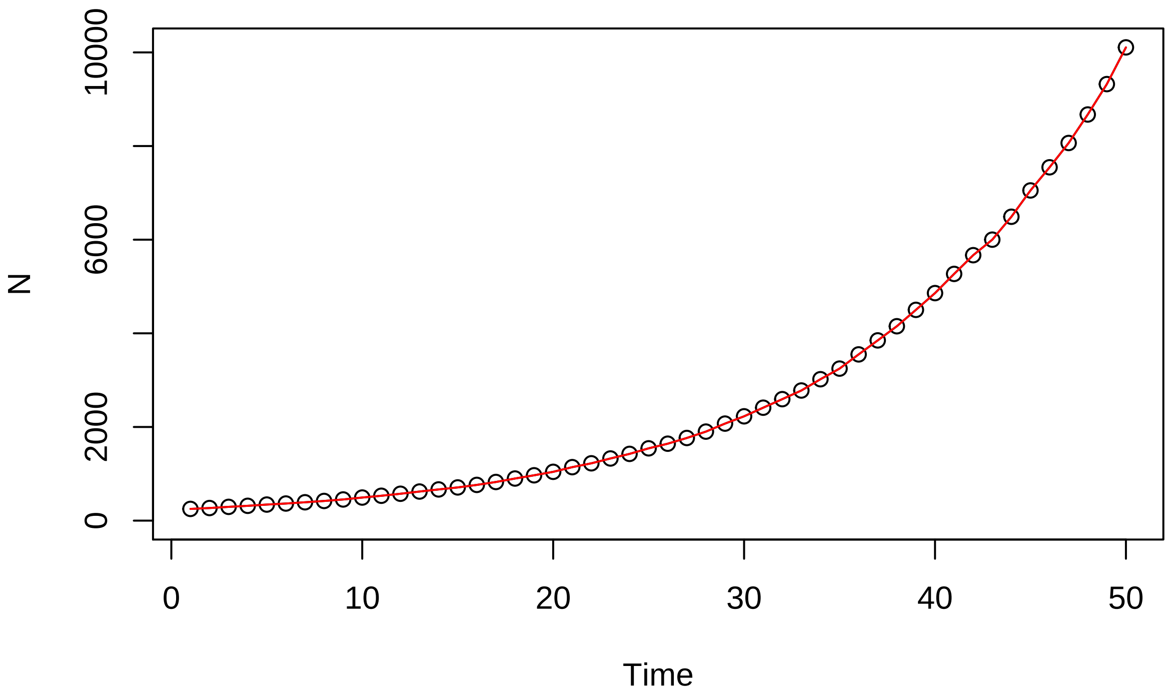
```r
# adding a density-independent migration component
```

```r
Model4 <- function(N_0=250,Time=50,r=0.08,sdR=0,lam=0){

  Nvec <- rep(0,Time) # create empty vector of length Time
  Nvec[1] <- N_0 # set initial value from input parameter
  Nimm <- rpois(n=Time,lambda=lam) # add random vector of immigrants
  r <- rnorm(n=Time,mean=r,sd=sdR) # create vector of random r values

    for (i in 2:Time) {
    Nvec[i] <- floor(Nvec[i-1] + r[i-1]*Nvec[i-1]) + Nimm[i-1]
    if(Nvec[i]<1){
      Nvec[i] <- 0  #no break because migrants can revive the population!

    }
    }
plot(x=1:length(Nvec),y=Nvec,type="o",xlab="Time",ylab="N",ylim=c(0,max(Nvec)))
  return(Nvec)

}
Nvec <- Model4(r=0.08,sdR=0.01)
```




```r
# adding measurement error
```

```
Model5 <- function(N_0=250,Time=50,r=0.08,sdR=0,lam=0,meas=0){

  Nvec <- rep(0,Time) # create empty vector of length Time
  Nvec[1] <- N_0  # set initial value from input parameter
  Noise <- round(rnorm(n=Time,mean=0,sd=meas*100)) # create vector of measurement error
  Nimm <- rpois(n=Time,lambda=lam) # create vector of random immigrants
  r <- rnorm(n=Time,mean=r,sd=sdR) # create vector of random r values

    for (i in 2:Time) {
    Nvec[i] <- floor(Nvec[i-1] + r[i-1]*Nvec[i-1]) + Nimm[i-1]
    if(Nvec[i]<0){
      Nvec[i] <- 0 # no break if immigrants allowed!
    }
    }
plot(x=1:length(Nvec),y=Nvec,type="o",xlab="Time",ylab="N",ylim=c(0,max(Nvec+Noise)))

# add the measurement error
Nvec <- Nvec + Noise
Nvec[Nvec<0] <- 0 # reset negatives to 0

# add the observed data points (red line) to the plot of true values

points(x=1:length(Nvec),y=Nvec,type="l",col="red")
  return(Nvec)

}
Nvec <- Model5(r=0.08,sdR=0.01)
```



N
0
2000
6000
10000
0
10
20
30
40
50
Time